Identifying Persistent X States in Serving RAM's Read Path

This analysis focuses on an early-stage waveform captured during simulation of an SRAM module with a Wishbone interface. The waveform highlights a critical issue: persistent X (undefined) values on the `o_rdata` signal for multiple cycles during read operations. This behavior can lead to non-deterministic data reads and violates expected timing behavior for memory access. By examining this waveform, we pinpoint the underlying causes and their ripple effects on the bus signaling and output data.

Phase-I:

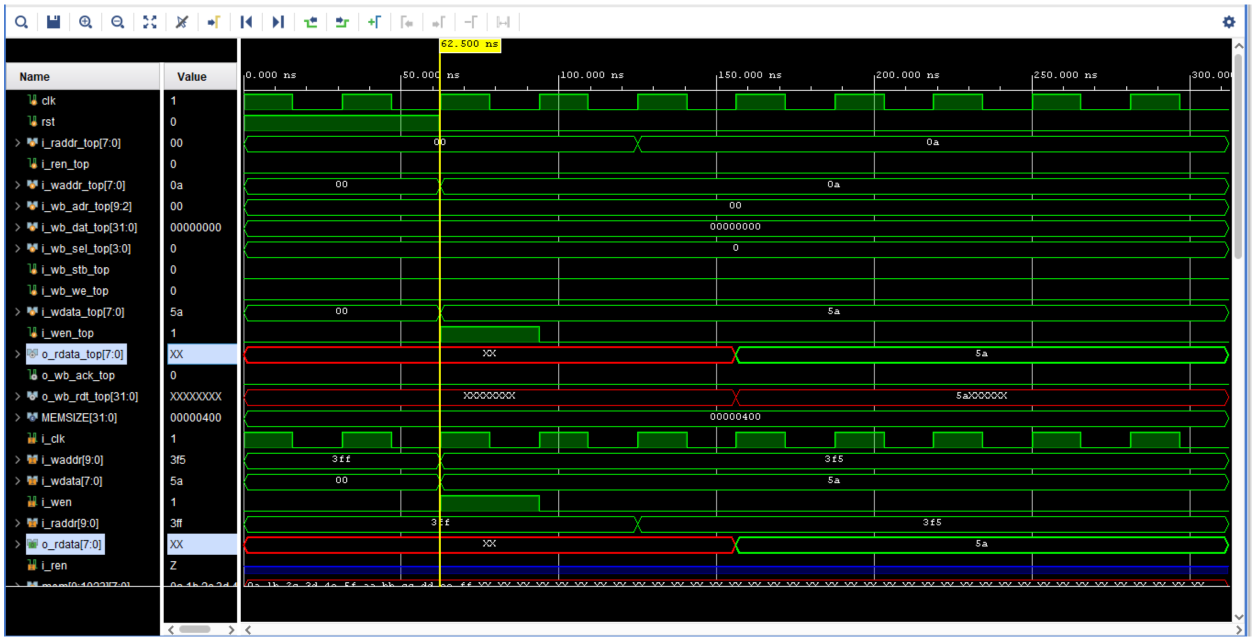

Figure: 01 – Faulty Read Behavior – Baseline Waveform for Debugging

Analysis of the Waveform (Image Above)

This waveform captures faulty behavior in the SRAM's read path. Specifically:

→ **`o_rdata_top[7:0]` and `o_rdata[7:0]` stay X for an extended duration**, even after valid address and enable signals are applied.

→ A read request is made (`i_ren_top = 1, i_raddr_top = 0x00`) while the system is out of reset, yet the read data output is not valid for **multiple clock cycles**.

→ o_rdata is not driven immediately and stays undefined. This suggests either the internal memory was not initialized or the logic to drive o_rdata was gated behind handshake signals.

→ Meanwhile `i_wdata_top, i_waddr_top`, etc. show correct behavior, confirming the write path works fine.

Core Problem

Figure 01 represents a **functional issue** — not just a transient glitch. The prolonged undefined state of o_rdata makes it unreliable during the first few read cycles. The root cause lies in the lack of:

→ Proper memory initialization,

→ Default assignments to output signals,

→ Immediate data forwarding logic upon valid reads.

This behavior must be corrected to ensure timing-accurate and deterministic reads from the memory module.

Phase-II:

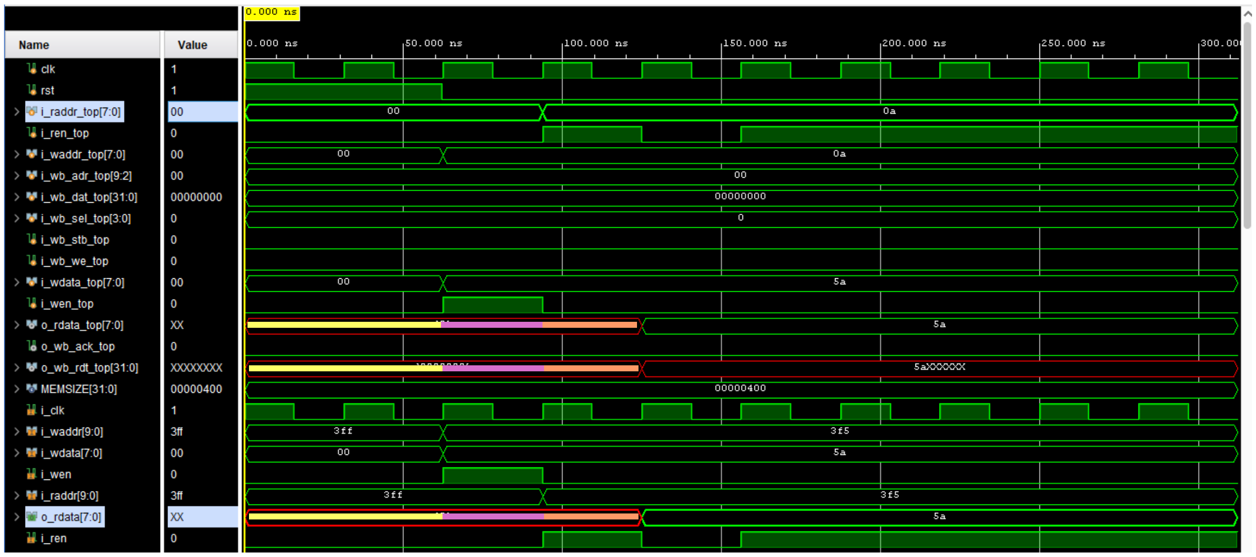

Figure: 02 – Regions of fault identified

🟨 Yellow Region: Reset is High

In the yellow-highlighted segment at the beginning of the simulation, the `i_rst` signal is asserted (1). During this phase:

→ Although the clock is toggling, the design is **logically in reset**.

→ Even if the `serving_ram` module does **not contain explicit reset logic**, it is being driven by signals from `servile_rf_mem_if` which is sensitive to reset.

→ Since the memory isn't being read or written (due to the system still in reset), and because o_rdata is not explicitly assigned during reset, it remains **undefined (X)**.

🟪 **Pink Region: Write Enable (i_wen) is High**

In this highlighted segment, you observe that i_wen is high, indicating a **write operation** is taking place. During this time:

→ The RAM is writing i_wdata to the location specified by i_waddr.

→ However, in the original implementation of serving_ram, the o_rdata register is **only updated when i_wen is low** (i.e., during a read).

→ Since o_rdata is not assigned during writes, its value **lingers in an undefined state** unless previously defined.

🟧 **Orange Region: Address Setup Violation**

This region highlights a **timing issue** — the read address i_raddr is not stable before the rising edge of i_clk.

→ RAM typically requires **address setup time**, meaning the address should be stable for a short duration **before the clock edge**.

→ In this case, i_raddr is changing **exactly at or just before** the rising edge.

→ This means the value latched by the RAM could be unpredictable or lead to undefined behavior.

→ The result is that the mem[i_raddr] access in the RAM returns an X, which is then assigned to o_rdata.

Phase-III:

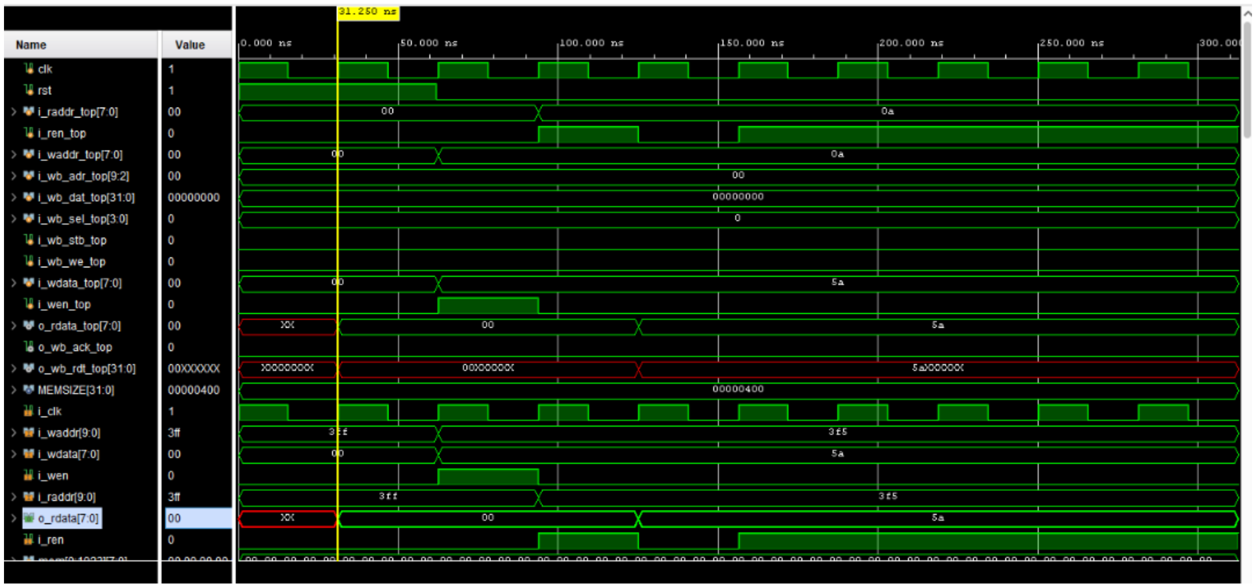

Figure: 03 – Response after memory initialization

```
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '1' seconds
Time resolution is 1 ps
Running RAM translation/access test...
[READ TEST] Addr: 0x0a, Expected: 0x5A, Got: 0x5a
```

Initially, o_rdata is undefined (X) because it hasn't been assigned a value before the first clock edge. After the rising edge of i_clk and deassertion of i_rst, a write occurs (i_wen=1, writing 0x5a to address 0x3f). On the subsequent read (i_wen=0), o_rdata correctly outputs 0x5a, confirming successful memory write and readback.

Phase-IV:

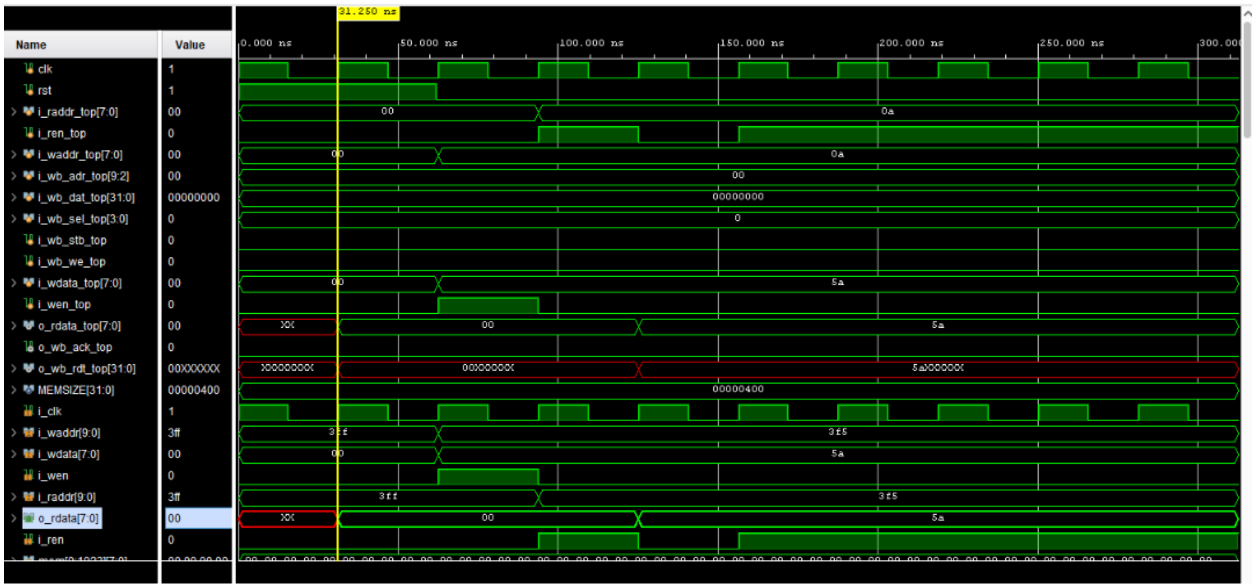

Figure: 04 – X during first clock cycle

Observation

During simulation of the top module, the output signal o_rdata displays an unexpected behavior:

→ o_rdata is explicitly initialized to 8'h00 in the initial block.

→ However, in the waveform, **the value of o_rdata becomes X (undefined) during the very first clock cycle**.

→ Afterward, the value stabilizes to valid data.

This behavior is misleading, especially considering that both the memory (mem) and o_rdata were initialized to 0.

The Problem

Even though o_rdata is initialized as follows:

```
initial begin
  o_rdata = 8'h00;
  for (i = 0; i < depth; i = i + 1)
    mem[i] = 8'h00;
end
```

It turns into X on the first rising edge of i_clk.

→ Initialization did occur correctly at simulation time zero.

→ Memory contents were all zero (mem[i] = 8'h00).

→ Input signals like i_ren, i_raddr seemed to be valid.

Yet, o_rdata was not valid during the first active cycle.

Reason:

1. initial Block is overridden by the Always Block

The initial block sets `o_rdata = 0` **before any clock activity begins**. But once the simulation reaches the **first posedge i_clk**, the `always @(posedge i_clk)` block takes over.

If this always block does not assign a value to o_rdata **on that first clock edge**, then:

→ o_rdata now has a new driver (from the always block), but no value is assigned.

→ As a result, Verilog simulators interpret it as **undefined (X)**.

2. Missing Reset Assignment

Initially, the always block looked like:

```
always @(posedge i_clk) begin
  if (i_wen)
    mem[i_waddr] <= i_wdata;
  else
    o_rdata <= mem[i_raddr];
end
```

Now if:

i_wen = 1 during the first clock edge, the if (i_wen) condition will be true. The write operation will occur, but **`o_rdata` is not assigned at all** in this cycle.

As a result:

The simulator detects that o_rdata has **no assigned value from the always block**. Since initial values are overridden by procedural assignments once simulation starts, o_rdata becomes X.

The Solution

To ensure o_rdata never becomes undefined:

1. Reset Sensitivity:

Code was updated to:

```
always @(posedge i_clk) begin
if (i_rst)
    o_rdata <= 8'h00;
    if (i_wen)
     mem[i_waddr] <= i_wdata;
    else
     o_rdata <= mem[i_raddr];
end
```

This ensures that o_rdata is assigned a **known value (0)** at the first clock edge when reset is active.

Result:

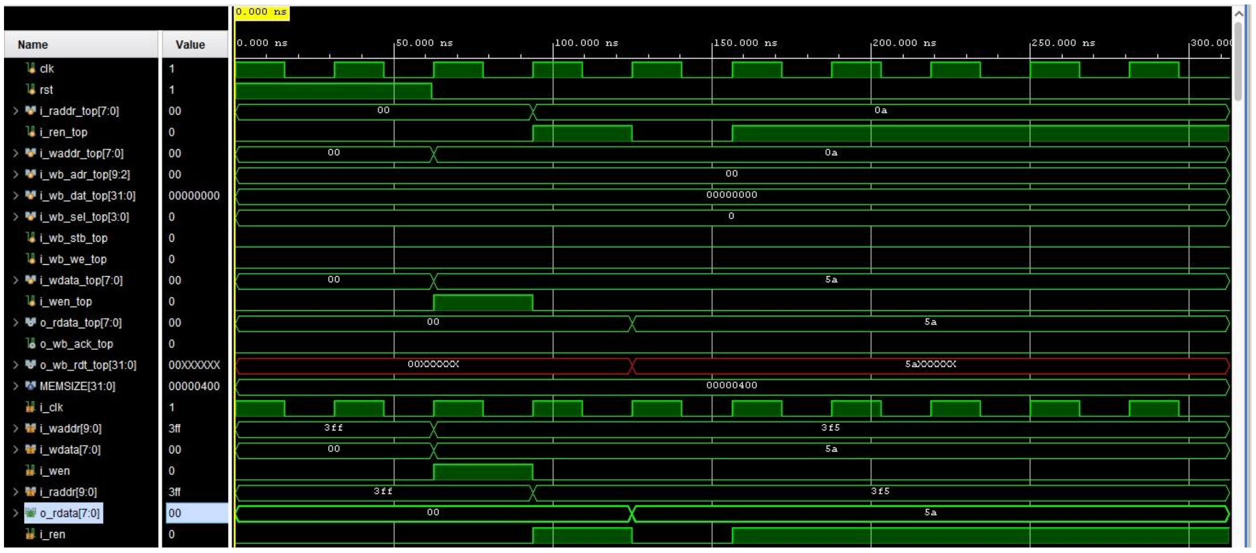

Figure: 04 – No undefined values in o_rdata (RAM output)